

METHOD OF LOCAL MINIMIZATION OF NONLINEAR MULTIDIMENSIONAL TARGET FUNCTIONS

1. Introduction

The nonlinear multidimensional target functions are known to be applied in optimization tasks. There are many methods of minimization (or maximization) of such functions. The methods that do not require calculating derivatives as well as do not require the use of empirical parameters deserve special attention for obvious reasons.

The method generated in this work belongs to such.

The method was tested by a test target function (11). Comparison of the result of this test with test results of other known methods has revealed larger speed of this method. That is, a smaller number of calls of target function for a given accuracy is achieved. This is effective in the cyclic operations in real time, for example, in controllers. Reducing of machine code volume and simplifying of initialization for cycle launch are other advantages.

Also the method was tested by ill-conditioned (poorly conditioned) target test functions of Powell (12) and Rosenbrock (13), and it worked too.

2. Designation agreements

Let there be an arbitrary function $f(x)$. Let its arbitrary points be numbered in ascending order: $0, 1, 2, \dots, \dots$.

$x^{[i]}$ is argument at i -th point.

$f^{[i]} \equiv f(x^{[i]})$ is value of function at i -th point.

$\Delta x^{[i]} \equiv x^{[i+1]} - x^{[i]}$.

$f^{(1)[i]} \equiv \frac{f^{[i+1]} - f^{[i]}}{x^{[i+1]} - x^{[i]}} = \frac{f^{[i+1]} - f^{[i]}}{\Delta x^{[i]}}$ is a divided difference of the 1st order of value at $[i+1]$ -th point relative to value at i -th point.

$f^{(2)[i]} \equiv \frac{f^{(1)[i+1]} - f^{(1)[i]}}{x^{[i+2]} - x^{[i]}} = \frac{f^{(1)[i+1]} - f^{(1)[i]}}{\Delta x^{[i]} + \Delta x^{[i+1]}}$ is a divided difference of the 2nd order of value at $[i+2]$ -th point relative to value at i -th point.

And in general:

$$f^{(j)[i]} \equiv \frac{f^{(j-1)[i+1]} - f^{(j-1)[i]}}{x^{[i+j]} - x^{[i]}} = \frac{f^{(j-1)[i+1]} - f^{(j-1)[i]}}{\Delta x^{[i]} + \Delta x^{[i+1]} + \dots + \Delta x^{[i+j-1]}} = \frac{f^{(j-1)[i+1]} - f^{(j-1)[i]}}{\sum_{l=i}^{i+j-1} \Delta x^{[l+1]}}$$

is divided difference of the j -th order of value at $[i+j]$ -th point relative to value at i -th point.

3. Generation of the method

3.1. Preliminary conclusions and how the method works

The basis of the method is the approximation of the target function by the Newton interpolation polynomial which is calculated (generated) on the basis of the values at the points obtained in previous iterations. The minimum of this polynomial is taken as point of the next iteration. When trying to find the minimum, you should find the extremum of this polynomial. To do this, its derivative should be equated to zero and the root(s) of this derivative should be found. How to find exactly the minimum will be shown below. In general, the degree of polynomial can be any but the 2nd degree is enough for iterations..

The book [1] provides the generation of Newton's polynomial for unevenly spaced points. There, the free member of the polynomial is the value of the function at the start point from a series of arbitrary points taken to generate the polynomial. And the i -th degree polynomial has the form (notations hereinafter - according to paragraph 2):

$$f(x) \approx f^{[0]} + f^{(1)[0]} \cdot (x - x^{[0]}) + f^{(2)[0]} \cdot (x - x^{[0]}) \cdot (x - x^{[1]}) + \dots + f^{(i)[0]} \cdot (x - x^{[0]}) \cdot (x - x^{[1]}) \cdot \dots \cdot (x - x^{[i-1]}).$$

But for greater accuracy of calculations when performing iterations, we need the free member of the polynomial to be the value $f^{[i]}$ of the function at the end such point, not $f^{[0]}$ at the start one. Let us generate the Newton polynomial as in [1] but for this case.

Based on the agreements of paragraph 2, we write:

$$\begin{aligned} f^{[i+1]} &= f^{(1)[i]} \cdot (x^{[i+1]} - x^{[i]}) + f^{[i]}; \\ f^{(1)[i]} &= f^{(2)[i-1]} \cdot (x^{[i+1]} - x^{[i-1]}) + f^{(1)[i-1]}; \\ f^{(2)[i-1]} &= f^{(3)[i-2]} \cdot (x^{[i+1]} - x^{[i-2]}) + f^{(2)[i-2]}; \\ f^{(3)[i-2]} &= f^{(4)[i-3]} \cdot (x^{[i+1]} - x^{[i-3]}) + f^{(3)[i-3]}; \\ &\dots; \\ f^{(j)[i-j+1]} &= f^{(j+1)[i-j]} \cdot (x^{[i+1]} - x^{[i-j]}) + f^{(j)[i-j]}; \\ &\dots; \\ f^{(i-2)[3]} &= f^{(i-1)[2]} \cdot (x^{[i+1]} - x^{[2]}) + f^{(i-2)[2]}; \\ f^{(i-1)[2]} &= f^{(i)[1]} \cdot (x^{[i+1]} - x^{[1]}) + f^{(i-1)[1]}; \\ f^{(i)[1]} &= f^{(i+1)[0]} \cdot (x^{[i+1]} - x^{[0]}) + f^{(i)[0]}. \end{aligned} \tag{1}$$

Implying $f^{[i+1]}$ as $f(x)$, i.e. as the value of current point of the polynomial and intending to find it, we insert the last expression into the previous one. The result we then insert into more previous one. And so on as shown below:

$$\begin{aligned} f^{(i)[1]} &= f^{(i+1)[0]} \cdot (x^{[i+1]} - x^{[0]}) + f^{(i)[0]}; \\ f^{(i-1)[2]} &= (f^{(i+1)[0]} \cdot (x^{[i+1]} - x^{[0]}) + f^{(i)[0]}) \cdot (x^{[i+1]} - x^{[1]}) + f^{(i-1)[1]}; \\ f^{(i-2)[3]} &= ((f^{(i+1)[0]} \cdot (x^{[i+1]} - x^{[0]}) + f^{(i)[0]}) \cdot (x^{[i+1]} - x^{[1]}) + f^{(i-1)[1]}) \cdot (x^{[i+1]} - x^{[2]}) + f^{(i-2)[2]}; \\ &\dots \end{aligned}$$

$$f^{[j+1]} = ((((((f^{(j+1)[0]} \cdot (x^{[j+1]} - x^{[0]}) + f^{(j)[0]}) \cdot (x^{[j+1]} - x^{[1]}) + f^{(j-1)[1]}) \cdot \dots + f^{(j+1)[j-1]}) \cdot (x^{[j+1]} - x^{[j-1]}) + f^{(j)[j-1]}) \cdot \dots + f^{(4)[j-4]}) \cdot (x^{[j+1]} - x^{[j-3]}) + f^{(3)[j-3]}) \cdot (x^{[j+1]} - x^{[j-2]}) + f^{(2)[j-2]}) \cdot (x^{[j+1]} - x^{[j-1]}) + f^{(1)[j-1]}) \cdot (x^{[j+1]} - x^{[j]}) + f^{[j]} \quad (2)$$

$$f^{(j+1)[0]} \equiv \frac{f^{(j)[1]} - f^{(j)[0]}}{x^{[j+1]} - x^{[0]}}$$

It is impossible to find the value because it is impossible to find the $f^{(1)[j]}$ value which is inserted there, because there is inserted $f^{[j+1]}$ which we actually are looking for.

Therefore, we assume that $f^{(j+1)[0]}$ is so small that it can be neglected, i.e. $f^{(j+1)[0]} \approx 0$. Naturally, the polynomial then becomes approximate, but it will be more accurate the greater its degree, i.e. the more points on the target function are taken to determine it.

This is the basis.

3.2. Finding the local minimum of a one-dimensional function

Let 3 points are known on some functions. From polynomial (2), being limited to 3 members of decomposition, we have:

$$f(x) \equiv f^{[j+1]} \approx (f^{(2)[j-2]} \cdot (x^{[j+1]} - x^{[j-1]}) + f^{(1)[j-1]} \cdot (x^{[j+1]} - x^{[j]}) + f^{[j]}).$$

We will only to approximate the target function using this polynomial (graphically, it is a parabola), and in this case we can replace the sign of approximation by a sign of equality. The extremum of the polynomial we find equating to zero its derivative, taken by $x^{[j+1]}$ (i.e., having adopted that $x^{[j+1]} = x$):

$$(f^{[j+1]})' = f^{(2)[j-2]} \cdot (2 \cdot x^{[j+1]} - x^{[j-1]} - x^{[j]}) + f^{(1)[j-1]} := 0, \quad (3)$$

where from

$$\begin{aligned} x^{[j+1]} &= \frac{-\frac{f^{(1)[j-1]}}{f^{(2)[j-2]} + x^{[j]} + x^{[j-1]}} - \frac{f^{(1)[j-1]} \cdot (x^{[j]} - x^{[j-2]})}{f^{(1)[j-1]} - f^{(1)[j-2]} + x^{[j]} + x^{[j]} - \Delta x^{[j-1]}}}{2} = \\ &= \frac{-f^{(1)[j-1]} \cdot \frac{\Delta x^{[j-2]} + \Delta x^{[j-1]}}{f^{(1)[j-1]} - f^{(1)[j-2]}} - \Delta x^{[j-1]}}{2} + x^{[j]}. \end{aligned}$$

We are interested for point of exactly minimum where $f''(x) > 0$. For this, we differentiate (3) relative to $x^{[j+1]} \equiv x$ and, given the agreements from paragraph 2, we have got:

$$\begin{aligned} (f^{[j+1]})'' &\equiv (f(x))'' = ((f^{(2)[j-2]} \cdot (2 \cdot x))' + (\frac{f^{(1)[j+1-2]} - f^{(1)[j-2]}}{x^{[j+2-2]} - x^{[j-2]}}))' = \\ &= (\frac{f^{(1)[j-1]} - f^{(1)[j-2]}}{x^{[j]} - x^{[j-2]}} \cdot (2 \cdot x))' = (\frac{f^{(1)[j-1]} - f^{(1)[j-2]}}{x^{[j]} - x^{[j-2]}})' \cdot (2 \cdot x) + \\ &= (\frac{f^{(1)[j-1]} - f^{(1)[j-2]}}{x^{[j]} - x^{[j-2]}}) \cdot (2 \cdot x)' = 0 + \frac{f^{(1)[j-1]} - f^{(1)[j-2]}}{x^{[j]} - x^{[j-2]}} \cdot 2 = 2 \cdot f^{(2)[j-2]} \end{aligned}$$

I.e. the condition $(f^{[i+1]})' > 0$ will be observed with such where $f^{(2)[i-2]} > 0$. Other x are not interested for us. Therefore, we take $f^{(2)[i-2]}$ in absolute value with a positive sign in front of this absolute value (and to find the maximum point, with a negative sign):

$$x^{[i+1]} = x^{[i]} + \Delta x^{[i]} = x^{[i]} + \frac{-\frac{f^{(1)[i-1]}}{f^{(2)[i-2]}} - \Delta x^{[i-1]}}{2} \quad (4)$$

or

$$\Delta x^{[i]} = \frac{-f^{(1)[i-1]} \cdot \left| \frac{\Delta x^{[i-2]} + \Delta x^{[i-1]}}{f^{(1)[i-1]} - f^{(1)[i-2]}} \right| - \Delta x^{[i-1]}}{2} \quad (5)$$

Formula (5) may be inserted in an iterative cycle of a computer program for finding a minimum of one-dimensional function.

3.3. Finding the local minimum of a n -dimensional function

n -dimensional target function is a scalar function of the vector argument. For approximation of such functions, Newton's interpolation polynomial in vector form is needed. It is possible to generate it according to the method similar to the above in clause 3.1 for a scalar form. Can be generated the any degree polynomial. Let's limit ourselves to the 2nd degree.

Let $x^{[i]}$ be a vector plotted at the i -th iteration from the origin to $x^{[i]}$ point in n -dimensional space. In this case

$f^{[i]} \equiv f(x^{[i]}) \equiv f(x^{[i]})$ is value of the function at this point.

$x^{[i]} = \sum_{k=1}^n x_k^{[i]} \cdot e_k$, where $x_k^{[i]}$ is a projection $x^{[i]}$ on the k -th coordinate; e_k is a single guide

vector (ort) according to the k -th coordinate;

$$\Delta x^{[i]} \equiv x^{[i+1]} - x^{[i]};$$

Let here be the concepts:

$f^{(1)[i]}$ is vector coefficient in the expression $f^{[i+1]} - f^{[i]} = f^{(1)[i]} \cdot (x^{[i+1]} - x^{[i]})$; (6)

$f^{(2)[i-1]}$ is coefficient in the expression $f^{(1)[i]} - f^{(1)[i-1]} = f^{(2)[i-1]} \cdot (x^{[i+1]} - x^{[i-1]})$; (7)

$f^{(3)[i-2]}$ is vector coefficient in the expression $f^{(2)[i-1]} - f^{(2)[i-2]} = f^{(3)[i-2]} \cdot (x^{[i+1]} - x^{[i-2]})$.

Taking $f^{[i+1]}$ as the current point $f(x)$ of the polynomial and setting out to find it we get:

$$\begin{aligned} f^{[i+1]} &= f^{(1)[i]} \cdot (x^{[i+1]} - x^{[i]}) + f^{[i]} = (f^{(2)[i-1]} \cdot (x^{[i+1]} - x^{[i-1]}) + f^{(1)[i-1]} \cdot (x^{[i+1]} - x^{[i]}) + f^{[i]} = \\ &= ((f^{(3)[i-2]} \cdot (x^{[i+1]} - x^{[i-2]}) + f^{(2)[i-2]} \cdot (x^{[i+1]} - x^{[i-1]}) + f^{(1)[i-1]} \cdot (x^{[i+1]} - x^{[i]}) + f^{[i]} \end{aligned}$$

We consider expression $f^{(3)[i-2]}$ to be an expression of higher order of smallness and set it to zero. Thus

$$f^{[i+1]} \approx f^{(2)[i-2]} \cdot (x^{[i+1]} - x^{[i-1]}) + f^{(1)[i-1]} \cdot (x^{[i+1]} - x^{[i]}) + f^{[i]}$$

This is only approximation of our real function $f(x)$ so the approximation sign can be replaced with an equal sign.

This can be rewritten in the form:

$$\begin{aligned}
 f^{[j+1]} &= f^{(2)[j-2]} \cdot (x^{[j+1]} - x^{[j-1]}) \cdot (x^{[j+1]} - x^{[j]}) + f^{(1)[j-1]} \cdot (x^{[j+1]} - x^{[j]}) + f^{[j]} = f^{(2)[j-2]} \cdot \left(\sum_{k=1}^n x_k^{[j+1]} \cdot \mathbf{e}_k - \right. \\
 &\quad \left. - \sum_{k=1}^n x_k^{[j-1]} \cdot \mathbf{e}_k \right) \cdot \left(\sum_{k=1}^n x_k^{[j+1]} \cdot \mathbf{e}_k - \sum_{k=1}^n x_k^{[j]} \cdot \mathbf{e}_k \right) + f^{(1)[j-1]} \cdot \left(\sum_{k=1}^n x_k^{[j+1]} \cdot \mathbf{e}_k - x^{[j]} \right) + f^{[j]} = \\
 &= f^{(2)[j-2]} \cdot \left(\sum_{k=1}^n (x_k^{[j+1]})^2 - \sum_{k=1}^n x_k^{[j-1]} \cdot x_k^{[j+1]} - \sum_{k=1}^n x_k^{[j]} \cdot x_k^{[j+1]} + \sum_{k=1}^n x_k^{[j-1]} \cdot x_k^{[j]} \right) + \\
 &\quad + f^{(1)[j-1]} \cdot \left(\sum_{k=1}^n x_k^{[j+1]} \cdot \mathbf{e}_k - x^{[j]} \right) + f^{[j]}
 \end{aligned}$$

To find the extremum of the polynomial, it is necessary to differentiate this with respect to $x_k \equiv x_k^{[j+1]}$ of the each k -th dimension and equate the obtained derivatives to zero:

$$\begin{aligned}
 (f^{[j+1]})'_{x_k^{[j+1]}} &= f^{(2)[j-2]} \cdot (2 \cdot x_k^{[j+1]} - x_k^{[j-1]} - x_k^{[j]}) + \left(\sum_{j=1}^n f_j^{(1)[j-1]} \cdot \mathbf{e}_j \right) \cdot \mathbf{e}_k = \\
 &= f^{(2)[j-2]} \cdot (2 \cdot x_k^{[j+1]} - x_k^{[j-1]} - x_k^{[j]}) + f_k^{(1)[j-1]} := 0.
 \end{aligned} \tag{8}$$

At the minimum (not maximum) point, the 2nd derivative is greater than zero. Therefore, by differentiating these derivatives again with respect to $x_k^{[j+1]}$,

$$(f^{[j+1]})''_{x_k^{[j+1]}} = 2 \cdot f^{(2)[j-2]}$$

we are convinced that to find the minimum we need to require that $f^{(2)[j-1]} > 0$.

Therefore,

$$\begin{aligned}
 x_k^{[j+1]} &= \frac{-\frac{f_k^{(1)[j-1]}}{f^{(2)[j-2]}} + x_k^{[j-1]} + x_k^{[j]}}{2} = \frac{-\frac{f_k^{(1)[j-1]}}{f^{(2)[j-2]}} + x_k^{[j]} - x_k^{[j]} + x_k^{[j-1]} + x_k^{[j]}}{2} = \\
 &= \frac{-\frac{f_k^{(1)[j-1]}}{f^{(2)[j-2]}} + x_k^{[j]} - \Delta x_k^{[j-1]} + x_k^{[j]}}{2} = \frac{-\frac{f_k^{(1)[j-1]}}{f^{(2)[j-2]}} - \Delta x_k^{[j-1]}}{2} + x_k^{[j]}
 \end{aligned} \tag{9}$$

where $\Delta x_k^{[j-1]} = x_k^{[j]} - x_k^{[j-1]}$.

To determine $\left| f^{(2)[j-2]} \right|$, we take into account that

$$f^{(1)[j-1]} - f^{(1)[j-2]} = f^{(2)[j-2]} \cdot (x^{[j]} - x^{[j-2]}) = f^{(2)[j-2]} \cdot (\Delta x^{[j-2]} + \Delta x^{[j-1]}).$$

Therefore,

$$\begin{aligned}
 \|f^{(1)[j-1]} - f^{(1)[j-2]}\| &= \left| f^{(2)[j-2]} \right| \cdot \|\Delta x^{[j-2]} + \Delta x^{[j-1]}\|; \\
 \left| f^{(2)[j-2]} \right| &= \frac{\|f^{(1)[j-1]} - f^{(1)[j-2]}\|}{\|\Delta x^{[j-2]} + \Delta x^{[j-1]}\|} = \sqrt{\frac{\sum_{k=1}^n (f_k^{(1)[j-1]} - f_k^{(1)[j-2]})^2}{\sum_{k=1}^n (\Delta x_k^{[j-2]} + \Delta x_k^{[j-1]})^2}}
 \end{aligned} \tag{10}$$

Formula (9) with the inclusion of (10) is applicable in a computer program.

4. How the fragment of computer program might look like (abstract)

Here some variables are represented in vector form. In programming environments capable of working with vector values, these variables can be announced directly. In other cases, the operations with vectors are easy to write as operations with data arrays.

Parameters

$\varepsilon = 0,0001$; //Accuracy criterion. Compares with $\|\Delta \mathbf{x}^{[j-2]} + \Delta \mathbf{x}^{[j-1]}\|$.

$n = 2$; //Number of coordinates, i.e. number of arguments of the target function.

Variables

$\mathbf{x}$; //Vector of arguments of the target function.

$f^{[i]}$; //The value of the target function after the i -th iteration.

$f^{[i-1]}$; //The value of the target function after the $i-1$ iteration.

$f^{(1)[i-1]}$; //Vector coefficient in expression $f^{[i]} - f^{[i-1]} = f^{(1)[i-1]} \cdot (\mathbf{x}^{[i]} - \mathbf{x}^{[i-1]})$.

$f^{(1)[i-2]}$; //Vector coefficient in expression $f^{[i-1]} - f^{[i-2]} = f^{(1)[i-2]} \cdot (\mathbf{x}^{[i-1]} - \mathbf{x}^{[i-2]})$.

$\Delta \mathbf{x}^{[i]}$; //Vector difference $\mathbf{x}^{[i+1]} - \mathbf{x}^{[i]}$.

$\Delta \mathbf{x}^{[i-1]}$; //Vector difference $\mathbf{x}^{[i]} - \mathbf{x}^{[i-1]}$.

$\Delta \mathbf{x}^{[i-2]}$; //Vector difference $\mathbf{x}^{[i-1]} - \mathbf{x}^{[i-2]}$.

k ; //Coordinate number.

nrm ; //To store the value of $\|\Delta \mathbf{x}^{[j-2]} + \Delta \mathbf{x}^{[j-1]}\|$, used in particular for comparison with ε .

Initialization

$\mathbf{x} = [0,5; 0,5]$;

$\Delta \mathbf{x}^{[i-1]} = [0,5; 0,5]$;

$\Delta \mathbf{x} = 0$; $\Delta \mathbf{x}^{[i-2]} = 0$; $f^{[i-1]} = 0$; $f^{(1)[i-2]} = 0$.

Implementation

do

For $k = 1$ to n :

$\mathbf{x}_k = \mathbf{x}_k + \Delta \mathbf{x}_k^{[i-1]}$;

$f^{[i]} = \frac{\exp(x_1 + x_2)}{x_1 + x_2^2}$; //Target function!

$f_k^{(1)[i-1]} = \frac{f^{[i]} - f^{[i-1]}}{\Delta x_k^{[i-1]} + 10^{-100}}$; //The " 10^{-100} " value is needed to protect against division by

0.

Next;

$nrm = \|\Delta \mathbf{x}^{[i-2]} + \Delta \mathbf{x}^{[i-1]}\|$;

$$\Delta \mathbf{x}^{[i-2]} = \frac{-f^{(1)[i-1]} \cdot nrm}{\|f^{(1)[i-1]} - f^{(1)[i-2]}\| + 10^{-100}} - \Delta \mathbf{x}^{[i-1]}$$
;

$\Delta \mathbf{x}^{[i-2]} := \Delta \mathbf{x}^{[i-1]}$; $\Delta \mathbf{x}^{[i-1]} := \Delta \mathbf{x}^{[i]}$; $f^{(1)[i-2]} := f^{(1)[i-1]}$; //Next i .

while $nrm > \varepsilon$; //Exit the loop when the required accuracy is achieved.

5. Tests and comparison

The testing by function [2]

$$f(x) = \frac{\exp\left(\sum_{j=1}^n x_j\right)}{\prod_{j=1}^n x_j^j} \quad (11)$$

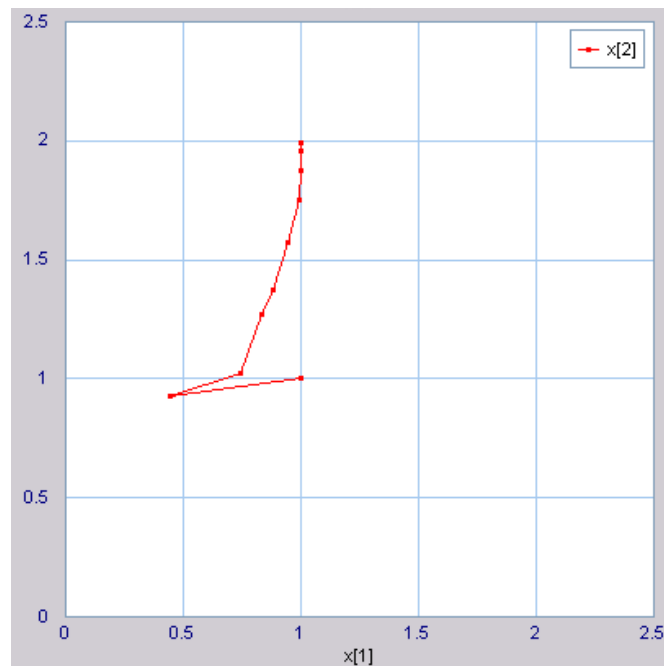
for $n = 2$ with initial values $x_1^{[0]} = x_2^{[0]} = 0,5$ и $\Delta x_1^{[0]} = \Delta x_2^{[0]} = 0,5$ and a analytically determined minimum $f^{[*]}(1,0000000000; 2,0000000000) = 5,02138423079692$ shows that in 17 complete iterations (under complete iteration here we mean one period of processing all n coordinates) , i.e. in 34 calls to the target function, the method is able to reach the value

$x = \begin{bmatrix} 0,99999999 & 963049 \\ 1,99999357 & 415238 \end{bmatrix}$ at which $f = 5,02138423 \ 084875$ with $\varepsilon = 0,0001$. And with $\varepsilon = 0,001$
 $x = \begin{bmatrix} 0,99987791 & 8171296 \\ 1,99998778 & 069913 \end{bmatrix}$ for 14 complete iterations, i.e. in 28 the function calls, with
 $f = 5,02138426 \ 840669$.

For comparison, some method for similar purpose, built into some well-known mathematical modeling program and using gradients of the 1st и 2nd order, in 47 iterations reaches the value

$x = \begin{bmatrix} 0,99999920 & 452505 \\ 2,00000385 & 59343 \end{bmatrix}$, at which $f = 5,02138423 \ 08172$.

Below is the trajectory of the method with target function (11) for $n = 2$, where the abscissa is x_1 and the ordinate is x_2 . The dots mark the coordinates after each complete iteration. The algorithm starts successfully when all algorithm variables except x and Δx are initially reset to zero.



The method was also tested on more difficult, ill-conditioned test functions specifically designed for testing such methods.

Powell's function test [3]

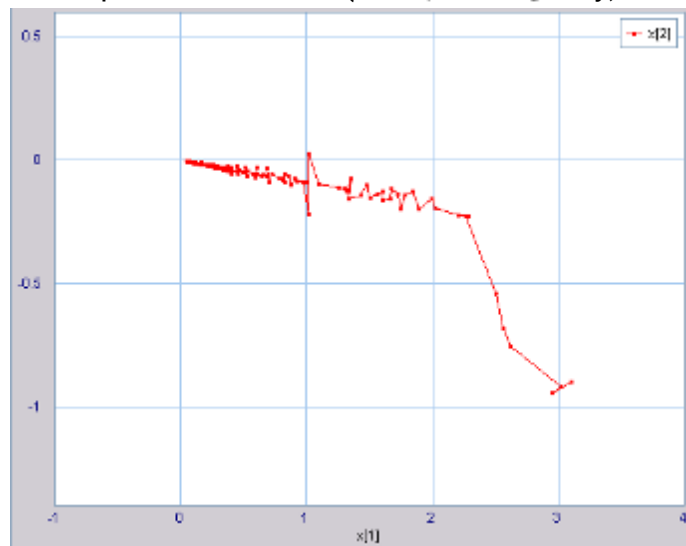
$$f(x) = (x_1 + 10 \cdot x_2)^2 + 5 \cdot (x_3 - x_4)^2 + (x_2 - 2 \cdot x_3)^4 + 10 \cdot (x_1 - x_4)^4, \quad (12)$$

which is not convex and its minimum point is degenerate, with $n = 4$, with initial values $\mathbf{x}^{[0]} = \begin{bmatrix} 3 \\ -1 \\ 0 \\ 1 \end{bmatrix}$

and $\Delta \mathbf{x}^{[0]} = \begin{bmatrix} 0,1 \\ 0,1 \\ 0,1 \\ 0,1 \end{bmatrix}$ and having an analytically determined minimum $\mathbf{x}^{[*]} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, shows that in 3736

complete iterations the method reaches the value $\mathbf{x} = \begin{bmatrix} 0,03521049 & 13727151 \\ -0,00352012 & 022129091 \\ 0,01749406 & 64789859 \\ 0,01752430 & 49114772 \end{bmatrix}$. The trajectory of

the method with this function is presented below (for x_1 and x_2 only).

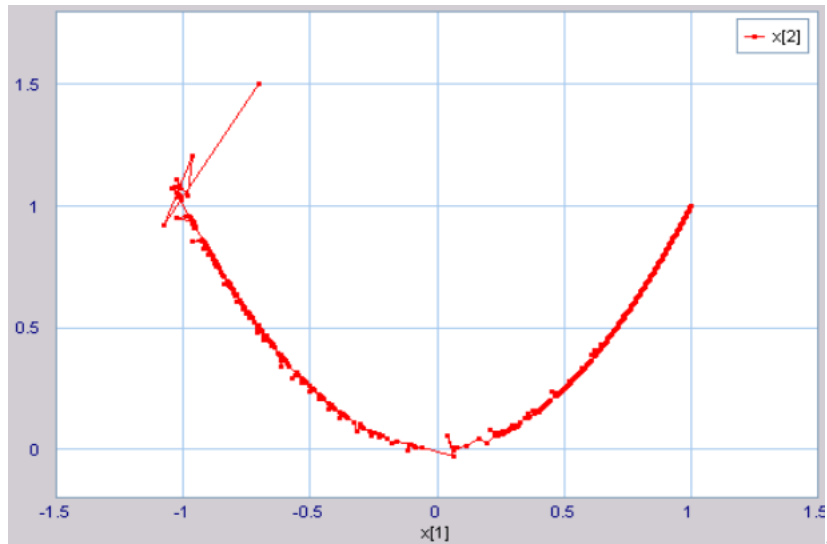


Test by the Rosenbrock function [3]

$$f(x) = 100 \cdot (x_2 - x_1^2)^2 + (1 - x_1)^2, \quad (13)$$

which is poorly conditioned, non-convex, with parabolic gully, with $n = 2$, with initial values $\mathbf{x}^{[0]} = \begin{bmatrix} -1,2 \\ 1 \end{bmatrix}$ and $\Delta \mathbf{x}^{[0]} = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$ and having an analytically determined minimum $\mathbf{x}^{[*]} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, shows

that in 7554 complete iterations the method reaches the values $\mathbf{x} = \begin{bmatrix} 0,99863970 & 006893 \\ 0,99727516 & 4636546 \end{bmatrix}$. Its trajectory is:



Литература

1. Крушевский А. В. и др. Вычислительная техника в инженерных и экономических расчетах. – Київ, «Вища школа», 1985.
2. Химмельблау Д. Прикладное нелинейное программирование. – М.: «Мир», 1976.
3. Поляк Б.Т. Введение в оптимизацию. – М.: «Наука», 1983.